**FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE**

# ASSIGNMENT OF BACHELOR'S THESIS

| | |
|---|---|
| **Title:** | Memory efficient cluster representations in non-metric spaces. |
| **Student:** | Jaroslav Hlaváč |
| **Supervisor:** | Ing. Martin Kopp |
| **Study Programme:** | Informatics |
| **Study Branch:** | Computer Security and Information technology |
| **Department:** | Department of Computer Systems |
| **Validity:** | Until the end of winter semester 2020/21 |

## Instructions

The goal of this thesis is to create a memory efficient representation of network host behavioural clusters used in the Cognitive targeted anomaly detection framework. The used behavioural similarity measure does not form a metric space. Therefore the non-metric cluster representation is needed.
Study the state-of-the-art literature on the topic of cluster representations in non-metric spaces.
Create benchmark datasets and use them to compare the memory and computational requirements of existing methods for cluster representations in non-metric spaces.
Analyze the results, select the best method, and incorporate it into the Cognitive targeted anomaly detection framework and fine-tune its parameters for the network security domain.

## References

Will be provided by the supervisor.

prof. Ing. Pavel Tvrdík, CSc.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague February 25, 2019

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

# Memory efficient cluster representations in non-metric spaces

***Jaroslav Hlaváč***

Department of Computer Systems
Supervisor: Ing. Martin Kopp

May 15, 2019

# Acknowledgements

I would like to express my thanks to my supervisor Ing. Martin Kopp for the inspiration I always brought from our meetings, for the opportunity to work on a real problem and last but not least for his patience.

# Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on May 15, 2019 .....

Czech Technical University in Prague
Faculty of Information Technology


### Citation of this thesis

Hlaváč, Jaroslav. *Memory efficient cluster representations in non-metric spaces*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

# Abstrakt

Internet je dnes nejpoužívanějším komunikačním médiem a proto je bezpečnost počítačových sítí aktuálním tématem. Monitorování opakujících se vzorů v chování jednotlivých sítí je jeden z možných přístupů k jejich zabezpečování. Přesnost a rychlost detekce anomálií je závislá na kvalitě modelu vytvořeného pro danou síť. Na základě chování jednotlivých prvků lze síť rozdělit na menší části. Tím se předejde příliš robustnímu modelu zanesenému šumem způsobeným různorodostí prvků v síti. Jeden z možných přístupů k vytvoření těchto částí je komunitní shlukování. Tyto shluky mohou mít až tisíce reprezentantů, pokud se jedná o velkou síť.

V této práci je prezentován algoritmus pro vytváření shlukových prototypů v Cognitive Targeted Anomaly Detection Framework. Momentálně jsou shluky v tomto frameworku reprezentovány náhodnou podmnožinou prvků z celého shluku. Pro shlukování se používá párová podobnost mezi jednotlivými prvky. Proto je potřeba využít medodu pro výběr reprezentantů, která je použitelná i v topologických prostorech. Modifikovaný algoritmus $\delta$-Medoids je představen jako nová metoda pro vytváření shlukových prototypů. Algoritmus $\delta$-Medoids Modified pro každý shluk zajišťuje výběr malého množství reprezentantů s maximálním množstvím zachované informace pro použití v klasifikačních problémech a je rychlejší než nemodifikovaná verze algoritmu.

**Klíčová slova** algoritmus $\delta$-Medoids, Detekce anomálií, Komunitní shlukování, Párová podobnost, Systém detekce narušení, Topologický prostor, Výběr reprezentantů

# Abstract

The Internet is a dominant medium for communication, therefore securing computer network is a crucial task. One approach for securing networks is monitoring their behavior patterns. Precise and fast network behavior anomaly detection highly depends on the models created for the monitored network. To create a model that is not clogged by noise from the whole network, hosts can be separated into smaller groups by community-based clustering. On big networks of tens of thousands devices, these clusters can contain several thousands of hosts.

This thesis presents an algorithm for finding cluster prototypes for clusters in Cognitive Targeted Anomaly Detection Framework. In the framework, a pair-wise similarity measure is used for clustering. Clusters are then represented by a number of randomly selected samples. Therefore, representative selection methods for topological spaces have to be used. A modification of $\delta$-Medoids algorithm is proposed with the ability to select a low number of representatives while maintaining the best possible coverage of the cluster. The introduced algorithm is also faster then the unmodified version of $\delta$-Medoids.

**Keywords** $\delta$-Medoids algorithm, Community-based clustering, Intrusion detection system, Network anomaly detection, Pair-wise similarity, Representative selection, Topological space

# Contents

# List of Figures

# List of Tables

# Introduction

The Internet became a dominant medium for communication several decades ago, and there are still many issues concerning its security. Companies all around the world connect more and more of their infrastructure to the Internet. Each connected device is an opportunity for attackers. Security engineers are always inventing new methods to stop the attacks and keep their networks safe and secure. It is a never-ending race between attackers, trying to overcome new security methods, and defenders, the security personnel, trying to be always one step ahead of the attackers. This race creates the drive for very sophisticated defense mechanisms.

Multi-layer defense systems are protecting contemporary networks. As attackers overcome one layer, there is yet another layer waiting for them. However, if the attackers succeed to bypass all of them unnoticed, a huge problem can occur. Attackers can collect user credentials, exfiltrate data or gain control of devices without anybody's knowledge. It can take a long time before the breach is noticed. A similar thing happened in Marriot International child company Starwood in 2018 [1]. Names, emails, addresses and credit card numbers of 500 million customers were stolen in an attack that was discovered in 2018 but possibly could have begun as early as in 2014.

One of several methods that are used to detect such intrusions is looking for anomalies in the behavior of network hosts - Network Behavioral Anomaly Detection (NBAD). In NBAD, devices inside the network are used as probes to gather information about the behavior of the network or individual hosts. The current behavior of the network is then compared to a model created from the historical data. If a significant deviation occurs, the security team is notified and can act accordingly.

It is not uncommon for a company to have tens of thousands of devices or even more. Using a single model for the whole network may be insufficient to detect breaches successfully. For example, an infection of a single device can be missed in the traffic of a medium sized company. In order to detect such anomalies, it is needed to focus on smaller parts of the network. Clustering

network hosts based on their behavior can be used to divide it into smaller pieces.

In Cognitive Targeted Anomaly Detection Framework from Cisco, host behavior is used to find groups of similar hosts in a network. The main focus of this thesis is to find a memory efficient representation of these clusters which also allows them to change dynamically in time. Four algorithms previously used to represent clusters were studied and modified for future use in Cognitive Targeted Anomaly Detection Framework. They were tested and compared on both real and test datasets.

The thesis is organized as follows. Chapter 1 introduces the anomaly-based network intrusion detection. Chapter 2 explains the theory behind clustering and representative selection in metric and non-metric spaces. Chapter 3 presents datasets that were used for testing. Chapter 4 covers all the experiments with their results and is followed by the conclusion of this thesis.

# Goals

This work aims to study methods that can be used in Cognitive Targeted Anomaly Detection Framework for the representation of network host behavioral clusters. Behavioral similarity used for clustering in this framework does not form a metric space, which creates a requirement for algorithms used for representative selection.

This thesis should present a method that can be used for finding the cluster prototypes of clusters in a non-metric space. Cluster prototypes represent the whole cluster in a way that is convenient for further computation, for example, assigning newly added samples to their corresponding cluster.

Networks monitored by Cognitive Targeted Anomaly Detection Framework can have up to several hundreds of thousands of hosts which leads to massive clusters. If the classification of newly added hosts were done by comparing the new host to all others, the memory and time needed for classification would be unmanageable. In the framework, each cluster is represented by a subset of its samples. The number of representatives selected from each cluster should be kept as small as possible to maintain computational and memory efficiency.

Methods used for general cluster representation in non-metric spaces should be studied. From these previously studied methods, the best-suited ones will be selected and implemented. As part of this thesis, a benchmark dataset for testing the chosen methods should also be created. A further goal is to test the chosen methods on clustering datasets used by others and on a newly created benchmark dataset. Based on this testing, a method should be selected that suits best the security field and can be incorporated into the Cognitive Targeted Anomaly Detection Framework. This method should be finetuned to get the best possible results in the framework.

Chapter 1

# Intrusion Detection System

This chapter presents the ideas from network security needed to understand the main objective of this thesis. Different approaches to network intrusion detection systems (IDS) are explained in Section 1.1. In Section 1.2, the scope is narrowed to network anomaly detection IDS and Section 1.3 delves deeper into the challenges of it. Section 1.4 serves as an introduction to Cognitive Targeted Anomaly Detection Framework used by Cisco. Methods researched in this thesis were tested on real data collected from Cognitive Targeted Anomaly Detection Framework.

## 1.1 Types of Intrusion Detection Systems

An intrusion detection system (IDS) is a security tool designed for identification of unauthorized use or abuse of computer systems by both system insiders and external penetrators [2]. IDS that is explicitly designed for monitoring computer network is called Network IDS (NIDS) as in comparison to host IDS and hybrid IDS. Host IDS focuses on monitoring the internal state of a computer system (e. g. mainframe computer) and its dynamic behavior. Hybrid IDS combines different techniques including the ones used in a host and network IDS creating an IDS that can leverage information from each of its parts for better intrusion detection. Examples of NIDS software are [3]:

- Snort
- Suricata
- Bro Network Security Monitor

These systems are used on computer networks to detect and or even prevent attacks that are threats to the essential services of such a network. These basic services are [2]:

- Data confidentiality

- Data and communication integrity
- Accessibility

Attackers try to disrupt these services by accessing confidential information (snooping), manipulating information (data tampering attacks) or disabling access to network services (Denial of Service attacks). IDS must have multiple components to detect as many of these attacks as possible. As each kind of intrusion is better detected by a different method, there are several types of IDS [4]:

- **Signature-based** (knowledge-based): Patterns of known attacks or threats are being compared to captured events for intrusion detection.
- **Anomaly-based** (behavior-based): A static or dynamic model of a given network is created over a period of time, and the current network behavior is compared to expected (model) behavior for anomalies.
- **Stateful protocol analysis** (specification-based): The system keeps track of known protocols (e. g., pairing requests with replies) and finding unexpected behavior in these protocols.

Cognitive Targeted Anomaly Detection Framework combines all of these approaches. This thesis focuses on the behavior-based part of the framework.

## 1.2 Network Anomaly Detection

Network anomaly detection is a method used in Intrusion Detection Systems (IDS) as explained in the previous section. This method focuses on comparing network host behavior changes in time. If a change more significant than a certain threshold is observed, network anomaly is detected. In IDS, when an anomaly is detected, an alert is created to inform the network administrator about what has happened. This alert can be any kind of message ranging from a syslog message to an email sent to the admin.

Host behavior is collected from devices in the network. There is at least one device (although many times multiple) dedicated for collection of network flows (using NetFlow protocol). Flow collection is the most widely used method for gathering data on the network. One network flow is an aggregated information about one connection that consists of source and destination addresses, source, and destination ports, begin and end timestamps for communication and size of data transferred in each direction [5]. This aggregation of information enables much faster (even real-time) detection of problems on a network as compared to, for example, deep packet inspection (DPI). DPI is another method used to detect problems in a network. It focuses on exploring the payload of each packet and is mostly considered unusable. Not only

the majority of traffic is encrypted, but it is also impossible to look into each packet because of the enormous amount of traffic in today's networks.

Network flows are collected and then sent via NetFlow protocol to one place where they are stored, and evaluated by detection algorithms. Network anomaly detection system creates a baseline model of the network behavior, which is then compared to the actual traffic. Any sudden change is considered to be an anomaly that is reported by the IDS.

## 1.3 Challenges of Real-Time Anomaly Detection

There are several challenges in anomaly-based network intrusion detection. A huge volume of data needs to be processed very fast to keep the real-time reaction rate to anomalies. Also, each normal behaviour that is identified as anomalous creates unwanted load for postprocessing of the occurence. Having a big amount of these false positives slows the process down. Finally there is a lack of labeled training data to create models of wanted network behavior [6].

Therefore, even having access to all the traffic in a network does not necessarily mean that every anomaly that occurs can be detected. Maintaining a model for each host is not an option as not every network host generates enough traffic to create a precise model of its behavior. Creating a behavioral model for the whole network is not a very good option either. The behavior of each host is different and combining them together leads to a very diverse model. A small amount of infected devices or a single intrusion attempt can be easily missed in this model due to the volume of information given by the rest of the network.

To give an example, imagine detection of a single infected device, such as a printer in a 10 story office building. This printer, being a part of a botnet, was ordered to generate traffic for DDoS (Distributed Denial of Service) attacks. Looking at the traffic of the printer before and after infection it could easily be seen that it increased by several hundred or even thousands of percent. However, an anomaly detection system could hardly notice a change when considering the traffic of the whole office building.

If an AD system does not mark malicious traffic as an anomaly, it is considered a false negative. That is another challenge AD systems are struggling with, to keep the number of missed attacks at 0 or as close to it as possible.

One more problem connected to false negatives are the false positives. A false positive is a network sample of benign network traffic marked as anomalous by the AD system. For example, if we would have our anomaly detection system set up wrong, it might detect an anomaly every day at 8 AM when workers come to the office, start their computers and download daily email. Comparing the time window from before 8 AM and after 8 AM would not give us any relevant information. This problem can be mitigated by setting a

window, for which the model is calculated, long enough so that these mistakes do not happen. There are many other situations when false positive alerts can happen. Having more than 0 false positives is not that big of a problem as having more than 0 false negatives. In IDS there is always a way to filter and double check the anomalies. However, a network anomaly detection system should avoid a big overhead in false positives as it could overwhelm the system.

In conclusion, a good anomaly detection system should detect all anomalies that are somehow connected to malicious behavior while trying to keep the number of false positives alerts as small as possible. The small number of false positives that are reported is then analyzed further in the following layers in the IDS.

## 1.4 Cognitive Targeted Anomaly Detection Framework

Algorithms studied in this thesis are tested as a part of Cognitive Targeted Anomaly Detection Framework which is a part of Cognitive Threat Analytics developed and used by Cisco. This framework successfully uses community-based clustering on the behavior of each host [7]. The aim is to split the whole network into smaller groups. Running anomaly detection for each group then yields significantly better results as compared to the traditional whole network approach. Not only it works better because of community-based clustering, but also because of the ability to adapt dynamically as the network changes. Thus it is able to incorporate changes that happen on a given network such as adding and removing devices.

The method is separated into two phases. In the initial phase, the state of the network is learned, and an initial model is created. Then, in the second ongoing phase, the framework dynamically adjusts clusters to the current state of the network.

The initial phase starts by collecting 24 hours of traffic from a given network. Collected data consists of network traffic flows and proxy server logs. Once the 24-hour period is over, clustering of hosts starts. Each host is represented by a tuple $h = S^h, F^h$, where:

- $S^h$ represents set of all visited pairs server:port
- $F^h$ represents the frequency of visits of server:port pairs

The frequency is defined as:

$$F_s^h = \frac{i}{n} \sum_{1}^{n} I(t_i, s, h) \,, \tag{1}$$

where $n$ is the number of time windows, $I$ is the indicator function, which is 1 if the network host $h$ visited the server $s$ in the timewindow $t_i$ and 0 otherwise. This ratio-based frequency ensures that frequently visited servers (e.g., Google, Facebook) do not overshadow the less frequently visited servers.

When each host is represented, the clustering algorithm is started. A technique called community-based clustering is used to create groups of hosts. This clustering works for graphs, where communities are more densely connected parts of the graph. In the words of this NBAD: Those hosts that communicate with similar peers are considered to be in the same community. Community-based clustering is explained in detail in Section 2.2.

The density of samples in sets of data is determined by cosine similarity of frequency vectors of two hosts. Similarity measure between hosts $a$ and $b$ is defined as:

$$\mathrm{sim}(a, b) = \frac{\sum\limits_{s \in S} F_s^a F_s^b}{\sqrt{\sum\limits_{s \in S} (F_s^a)^2} \sqrt{\sum\limits_{s \in S} (F_s^b)^2}} \,, \tag{2}$$

where $F^a$, $F^b$ represent the frequency and $S$ represents the union of sets of servers visited by the network hosts $a$ and $b$.

After the clustering is done, only clusters that are bigger than 10 hosts are selected for representative selection. Smaller clusters are analyzed using a fallback method - host-centric anomaly detection. For the purpose of this work, we do not consider clusters smaller than 10 hosts.

When clusters are determined, $\kappa$ random representatives are selected from each cluster. Currently, $\kappa$ is an empirically set parameter. In this thesis, tests were made to improve this random selection method. A more detailed explanation of this method can be found in [7].

After the initial training phase is over and the model is established, data continues to be collected. Every 4 hours all hosts that are observed are assigned to existing clusters. At that point, a portion of cluster representatives is replaced by new ones to capture the ever-changing nature of the network data.

These clusters are then further used in anomaly detection. They serve as a foundation for calculating baseline behavior for hosts belonging to it. If a host is known to belong to a cluster $A$ and its behavior suddenly starts to differ from the behavior of representatives selected for cluster $A$, an anomaly is found and reported further into the NIDS.

Chapter 2

# Theoretical Overview and State of the Art

This chapter introduces a theoretical background to explain clustering methods, the topic of non-metric spaces and the problem of representative selection.

## 2.1 Introduction to Clustering

As defined in [8], clustering is an unsupervised machine learning method that organizes objects into groups so that each group consists of members that are similar in some way. Without any prior knowledge of the data, this method looks for structures in feature vectors. For each cluster $c$, a variability can be calculated. It shows how much objects in given cluster differ. Variability and dissimilarity are two properties defined for a better understanding of the data. Variability is defined as

$$\text{variability}(c) = \sum_{e \in c} \text{distance}(\text{mean}(c), e)^2 \,, \tag{3}$$

where $e$ is an object from given cluster. The distance is a measure that quantifies the proximity of two objects with the same number of features. Many different distance measures are used in clustering. Examples of commonly used ones are:

- Euclidean distance
- Manhattan distance
- Mahanalobis distance
- Cosine similarity

Dissimilarity is defined as

$$\text{dissimilarity}(C) = \sum_{c \in C} \text{variability}(c)\,, \tag{4}$$

where $C$ stands for a set of all clusters. For clustering, the aim is to keep the dissimilarity of all clusters from the dataset as low as possible. Given this definition, the best way to cluster every dataset would be to put each object to its cluster. That would not lead to any reasonable result. Therefore there is a constraint added for clustering methods. It can be either the maximum number of clusters or the maximum distance between two clusters.

A straightforward example of clustering is a method called agglomerative hierarchical clustering [8]. Given $N$ objects in a dataset, it creates $N$ clusters - meaning there is a cluster for each object. The method looks for two closest clusters and merges them into one. This agglomerative merging continues until the constraint is met, meaning until there is a certain number of clusters or until the distance between closest clusters exceeds a certain threshold. This method is a greedy algorithm, and therefore it might not result in globally optimal clustering. Also, the algorithm has a time complexity of $\mathcal{O}(n^2)$. Therefore, it cannot be used in big datasets.

An example of a much faster clustering algorithm that is also greedy is $K$-means [8]. The '$K$' in $K$-means stands for the number of clusters that we want to get as a result. To use this algorithm, the number of desired clusters has to be known in advance. $K$-means randomly chooses $K$ centroids in the space of the dataset and then assigns each point to a centroid. After creating these clusters, it calculates a new centroid for each cluster and then assigns the points in datasets to the new centroids. The algorithm stops when the centroids of clusters stop changing. Algorithm 1 shows the pseudocode of $K$-means algorithm.

This algorithm is fast, it has time complexity $\mathcal{O}(k \cdot n)$, where $k$ is the number of clusters and $n$ is the number of objects in a dataset. It is the most common clustering algorithm as it typically converges in a few iterations. For more details about basic clustering algorithms see [8].

The clustering, as explained in this section has restrictions make it inaplicable in Cognitive Targeted Anomaly Detection Framework. Firstly, the number of clusters is not previously known, so choosing $K$ for $K$-means is not an option, and secondly, the distance measure used does not form a metric space. Next section delves deeper into what it means when a measure does not form a metric space.

## 2.2 Clustering in Topological Space

This section explains how does clustering approaches differ in topological spaces. A metric space is a topological space with special properties, that

**Algorithm 1** $K$-means

**Input:** data $X = x_0, x_1, ..., x_n$; number of clusters $k$
**Output:** $k$ clusters; $k$ centroids

$t = 0$
Initialize $centroids_t = k$ randomly chosen examples from $X$
**do**
  $t = t + 1$
  Initialize $clusters = \emptyset$
  **for** $c$ in $centroids_t$ **do**
    Initialize $cluster_c = \emptyset$
    add $cluster_c$ to $clusters$
  **end for**
  **for** $x$ in $X$ **do**
    $closest\_centroid = \text{argmin}_{c \in centroids} d(c, x)$
    add $x$ to $cluster_{closest\_centroid}$
  **end for**
  $centroids_t = \emptyset$
  **for** $cluster$ in $clusters_t$ **do**
    $new\_centroid = \text{mean}(cluster)$
    add $new\_centroid$ to $centroids_t$
  **end for**
**while** $centroids_t = centroids_{t-1}$
**return** $clusters$, $centroids_t$

are given in its definition. Therefore, each clustering algorithm that works on a topological space will work on a metric space also. However, there are algorithms that give better results in metric spaces that will not work by definition on a topological space, i.e. $K$-Means algorithm.

A metric space (see e.g. [9]) is a pair $(X, d)$ where $X$ is a set and $d$ is a mapping $X \times X \to \mathbb{R}$ which satisfies the following conditions:

(i) $d(x, y) \geq 0$;

(ii) $d(x, y) = 0 \iff x = y$;

(iii) $d(x, y) = d(y, x)$

(iv) $d(x, z) \leq d(x, y) + d(x, z)$ for $x, y, z \in X$.

Any function $d$ following these conditions is called distance.

The similarity measure in Equation 2 does not fulfill the last point of the definition above, as is explained in Section 1.4. Consequently, it is not a distance and does not form a metric space. Furthermore, it is a pairwise similarity that forms a subspace for comparing each pair of samples. This

is why that similarity measure forms a topological space, which is defined as follows (see e.g., [10]).

Given any set $S$ a topology on $S$ is a family $F = F_\alpha | \alpha \in A$, where $A$ is some indexing set, each $F_\alpha \subseteq S$, and with the following properties:

(i) The empty set $\emptyset$ is in $F$.

(ii) The given set $S$ is in $F$.

(iii) The intersection of any two sets of $F$ is in $F$.

(iv) The union of any number of sets of $F$ is in $F$.

The ordered pair $(S, F)$ is called a topological space.

$K$-means clustering from the previous section cannot be used, because centroids do not exist in topological spaces. A popular method that is used for clustering in non-metric spaces is called community-based clustering.

Community-based clustering detects communities in the data [7]. A community is a subset of examples in the data, that is densely connected with each other. One of the ways how to detect communities in a graph is to create a full-adjacency matrix. This matrix contains all connections between all nodes in the given graph. Analyzing the matrix can tell us about the densities in different parts of graphs.

Louvain method is a similar approach to clustering when we do not have a simple graph but a set of samples and a pairwise similarity measure [7]. This method relies on creating a full similarity matrix for the whole dataset and then looking for communities in the data.

In Cognitive Targeted Anomaly Detection Framework, Louvain method cannot be used directly as calculating the full similarity matrix has a complexity of $\mathcal{O}(n^2)$, which is impossible to calculate for a large network. That is why an approximative clustering method is used. This method iteratively samples network hosts and runs the clustering algorithm on the sampled hosts. Each iteration creates or updates cluster prototypes. If the data in the current batch fit into a previously prototyped cluster, they are added to it, and the cluster prototype is updated. If samples differ more than a predefined threshold, a new cluster prototype is created.

## 2.3 Representative Selection

This section focuses on the idea of finding a representation of clusters. Clustering huge datasets can result in big clusters of several tens of thousands of objects in them. Computational operations such as assigning new objects to clusters (e. g. $K$-Nearest Neighbors) are dependent on the number of objects in each cluster or the representation of these clusters. If it was possible to

represent these clusters in a different way than keeping all track of all of the objects, further operations on these clusters would run faster.

A cluster prototype is a data sample that represents all samples in the data cluster. According to [11], there are three motivations for finding the most representative cluster prototypes:

- Summarization
- Compression
- Efficient Finding Nearest Neighbors

All of these apply to the NBAD. Therefore, summarizing the behavior of the whole cluster into a cluster prototype is desirable. As is finding the smallest possible number of prototypes for each cluster for efficient finding nearest or most similar neighbors when adding new hosts to their corresponding clusters.

### 2.3.1 Definition of Representative Selection for Non-Metric Spaces

Representative selection aims to find a minimal subset of examples from a cluster, that carries sufficient information about the whole cluster. This problem was well defined in [12] and the following definition is taken from that paper.

Let $X$ be a data set, $d$: $X \times X \rightarrow \mathbb{R}+$ be a distance measure (not necessarily a metric), and $\delta$ be a distance threshold below which samples are considered sufficiently similar. The task is finding a representative subset $Z \subseteq S$ that best encapsulates the data. Two following requirements are imposed on an algorithm for finding a representative subset:

- **Requirement 1**: The algorithm must return a subset $Z \subseteq S$ such that for any sample $x \in S$, there exists a sample $z \in Z$ satisfying $d(x, z) \leq \delta$.
- **Requirement 2** : The algorithm cannot rely on a metric representation of the samples in $S$.

To compare the quality of different subsets returned by different algorithms, two criteria are measured:

- **Criterion 1**: $|Z|$ - seeking the smallest possible subset $Z$ that satisfies Requirement 1.
- **Criterion 2**: Representative should best fit the data on average. Given representative subsets of equal size, the preference is on the one that minimizes the average distance of samples from their respective representatives.

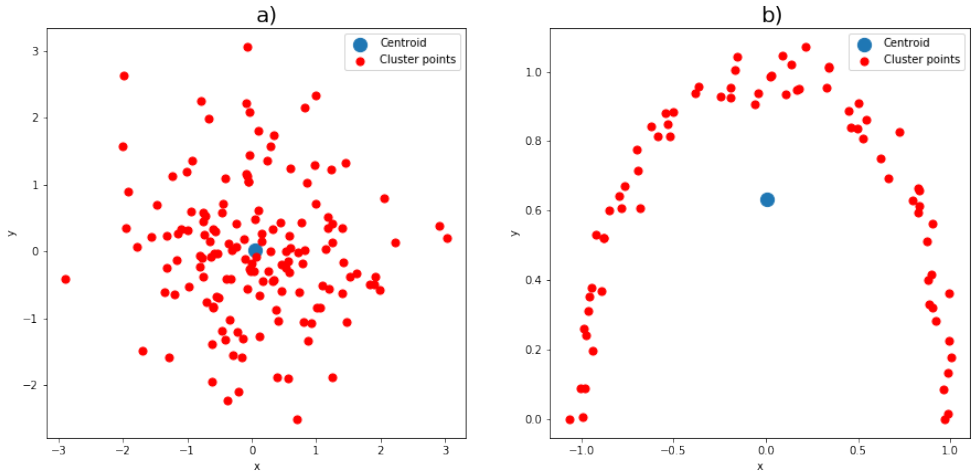


Figure 2.1: Centroids of a cluster calculated as mean of all points.

Criteria 1 and 2 are applied to a representative set solution. In addition, the following desiderata for a representative selection algorithm are expected.

- **Desideratum 1**: Stable representative selection algorithms are preferred. Let $Z_1$ and $Z_2$ be different representative subsets for dataset $S$ obtained by two different runs of the same algorithm. Stability is defined as the overlap $\frac{|Z_1 \cap Z_2|}{|Z_1 \cup Z_2|}$. The higher the expected overlap is, the more stable the algorithm is. This desideratum ensures the representative set is robust to randomization in data ordering or the choices made by the algorithm.

- **Desideratum 2**: The algorithm should be efficient and scale well for large datasets.

This definition of representative selection problem serves well for this paper.

### 2.3.2 Representative Selection in a Metric Space

A dataset that hase the same number of features for each sample and there is a metric metric that fulfills all the requisites in the definition of the metric space is a metric dataset. Selecting a prototype from this dataset is often best achieved by calculating a centroid for a given cluster. A centroid can be calculated as the mean of the points in the cluster. An example of a centroid in a cluster can be found in Figure 2.1 a). Other ways of calculating centroids can be used, e.g. weighted average of all points.

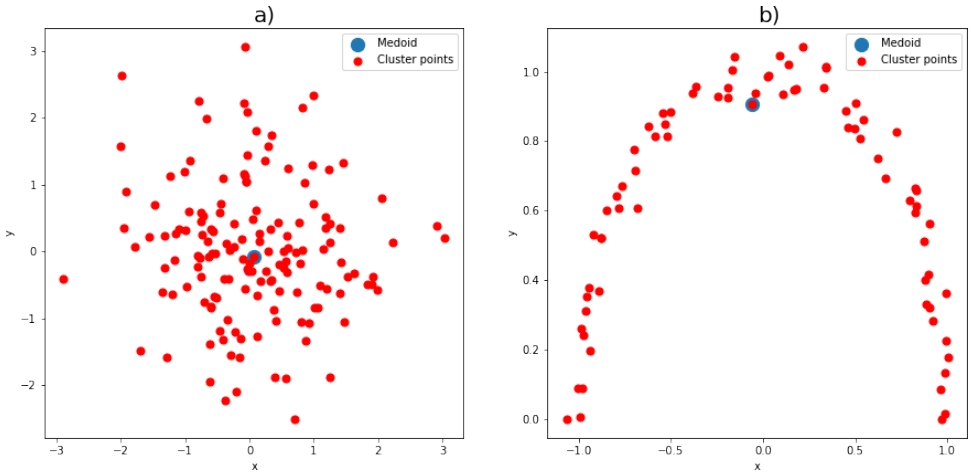


Figure 2.2: Choice of medoids

However, if a calculated centroid is not meaningful (see Figure 2.1 b)), a medoid can be selected as an alternative to it. A medoid is a point that is in the set that minimizes the average distance to all the other points in the set. It can be thought of as a median of the dataset. Formally, medoid is defined as:

$$x_{medoid} = \text{argmin}_{y \in \{x_1, x_2, ..., x_n\}} \sum_{i=1}^{n} d(y, x_i) \,, \tag{5}$$

where $x_1, x_2, ..., x_n$ is a set of $n$ points in a space with a distance function $d$. In Figure 2.2, medoids were chosen instead of centroids.

### 2.3.3 Representative Selection in a Topological Space

According to the previous section, the concept of centroid cannot be used to solve the problem of finding a representative in an arbitrary space (i.e., topological space). Instead, medoids can be selected for datasets in arbitrary spaces.

$K$-Medoids algorithm resembles $K$-means algorithm in breaking the dataset into $K$ groups. In these groups it finds a medoid that is minimizes the distance to each other point in the group. $K$-Medoids is most commonly used for representative selection in topological spaces.

There are not many other well-explored methods for solving the problem of finding medoids. The one that is used in this thesis is explained in [12]. The main ideas from $K$-means clustering were taken and transformed for usage in a non-metric space with a pair-wise similarity measure. Instead of stating the $K$ in advance they state a parameter $0 \leq \delta \leq 1$ that serves as a constraint. Then they separate the cluster into subclusters based on this parameter. Each

of these subclusters is represented by a medoid. A set of these medoids then serves as a cluster prototype for the whole cluster. This method is explained in greater detail in the Section 2.4.3.

## 2.4 Algorithms Relevant for Topological Space

Based on the previously explained approaches for finding representatives of clusters in non-metric spaces, the following methods were chosen, tested and compared as a part of this thesis.

- Random selection
- Greedy Selection
- $\delta$-Medoids One-Shot
- $\delta$-Medoids

### 2.4.1 Random Selection

Random selection algorithm serves as a baseline to measure the improvement as it is the method currently used in Cognitive Targeted Anomaly Detection Framework. It selects a given number of representatives from the cluster randomly.

In dense clusters, selecting random samples from each cluster leads to undesired noise. However as experimental results presented in [7] in the domain of computer networks, the density of clusters is not significant. It, of course, depends on each network, but given the best practices in segmentation of networks, the overlaps in clusters are not expected to be great.

### 2.4.2 Greedy Selection

The greedy selection algorithm is a straightforward algorithm that selects a random sample from the set as the first representative. Then it looks for the most different sample in the rest of the dataset. Once the desired sample is found, the algorithm removes all neighboring samples with at least the similarity of $\delta$ from the set. Then it looks for other dissimilar samples until the whole set is represented.

The speed of this algorithm depends on the sparsity of the given set. In the worst case scenario, all points would be selected as representatives while always calculating the distance to each remaining sample in the set. Then it would reach the time complexity of $\mathcal{O}(n^2)$.

### 2.4.3 $\delta$-Medoids

The two following algorithms were first explained in [12]. They were tested on two different problems that also use clustering in non-metric space - computing distance of two musical segments and comparing trajectories of objects.

$\delta$-Medoids algorithm tries to find a minimal subset of points that are needed to cover the cluster with only one given constraint, that distance of the representatives would be at least $\delta$. There are two versions of this algorithm. One-shot version is faster but less precise than the full $\delta$-medoids algorithm.

### 2.4.4 $\delta$-Medoids One-Shot

The idea behind this algorithm is to go through the dataset looking for representatives that differ at least by the given parameter $\delta$ from each previously selected representatives. By taking this approach, it can in one iteration over data find the representatives that are certain to differ by $\delta$ from each other.

The pseudocode of this algorithm is shown in Algorithm 2. It have been optimized for better memory efficiency as opposed to the version in the cited paper. The main ideas remain the same, only keeping track of clusters that the points are assigned to is removed for this one-shot version.

**Algorithm 2** $\delta$-Medoids One-shot

**Input:** data $x_0$ ... $x_m$, required distance $\delta$

```
Initialize representatives = ∅
for i = 0 to m do
    Initialize dist = ∞
    for rep in representatives do
        if d(x_i, rep) ≤ dist then
            dist = d(x_i, rep)
        end if
    end for
    if dist > δ then
        add x_i to representatives
    end if
end for
```

The choice of the first representative strongly influences the selection of the medoids to represent the cluster. In some cases, this one-shot approach could be misleading.

### 2.4.5 $\delta$-Medoids Full

The full version of the algorithm runs a one-shot algorithm multiple times. Each time it goes through the dataset it selects a better medoid than before. If two consecutive passes through the data do not change any medoid, the

algorithm stops. This algorithm is shown in Algorithm 3. Original algorithm from [12] lacks the routine *ReduceClusters* on line 24. This routine is introduced in this thesis as an improvement of the algorithm because the original algorithm selected too many representatives. It is explained in greater detail in the Section 2.4.6

This approach is slower than $\delta$-Medoids, as the main routine from one shot algorithm has to be run multiple times. Experimental results of [12] show that the algorithm converges in very few (<10) cycles. On the other hand, it is able to select better medoids, thus getting rid of the difficult choice of the first representative to select.

**Algorithm 3** $\delta$-Medoids

**Input:** data $x_0$ ... $x_m$, required distance $\delta$

1: $t = 0$
2: Initialize $representatives_{t_0} = \emptyset$
3: Initialize $clusters = \emptyset$
4: **do**
5: $t = t + 1$
6: **for** $i = 0$ **to** $m$ **do**
7: Initialize $dist = \infty$
8: Initialize *representative* = *null*
9: **for** *rep* in *representatives* **do**
10: **if** $d(x_i, rep) \leq dist$ **then**
11: *representative* = *rep*
12: $dist = d(x_i, rep)$
13: **end if**
14: **end for**
15: **if** $dist \leq \delta$ **then**
16: add $x_i$ to $cluster_{representative}$
17: **else**
18: $representative = x_i$
19: Initialize $cluster_{representative} = \emptyset$
20: add $x_i$ to $cluster_{representative}$
21: add $cluster_{representative}$ to *clusters*
22: **end if**
23: **end for**
24: Call ***ReduceClusters***
25: Initialize $representatives_t = \emptyset$
26: **for** *cluster* in *clusters* **do**
27: $representative = \text{argmin}_{s \in cluster}(\sum_{x \in cluster} d(x, s) : d(x, s) \leq \delta)$
28: add *representative* to $representatives_t$
29: **end for**
30: **while** $representatives_t = representatives_{t-1}$

### 2.4.6 $\delta$-Medoids Modified

The algorithm $\delta$-Medoids Full does not have any restriction for the number of representatives it selects for each cluster. Results in Chapter 4 show that the full algorithm can select up to a third of a big complex cluster which does not satisfy the Criterion 1 from Section 2.3.1 properly. To reduce the number of representatives a subroutine *ReduceClusters* was introduced as a modification to the original algorithm. The subroutine is shown *ReduceClusters* shown in Algorithm 4.

The basic idea behind this modification is to get rid of representatives that cover a small portion of the whole dataset by either merging them with similar representatives or dropping them entirely. If a representative does not cover at least 1% of the cluster, the routine tries to add it to the coverage of the representative that is the most similar. If no such similar representative exists, it is considered noise and the representative is dropped from the cluster.

**Algorithm 4** ReduceClusters

**Input:** $m$ subclusters $X = x_i, ..., x_m$; their representatives $R = r_0, ..., r_m$; whole cluster size$s$; constant $\kappa$

```
1:  threshold = 0.01s
2:  Initiate i = 0
3:  if |X| > κ then
4:    while i < m do
5:      if |x_i| ≤ threshold then
6:        j = index the most similar representative from R
7:        if no such j exists then
8:          Drop x_i from X
9:        else
10:         Merge x_i and x_j.
11:       end if
12:       Drop r_i from R
13:     end if
14:   end while
15: end if
```

Figure 2.3 shows that $\delta$-Medoids Modified is a bit faster than the version full version. This combined with the fact that it selects lower number of representatives shows that the modification improved its performance for this thesis's use case.

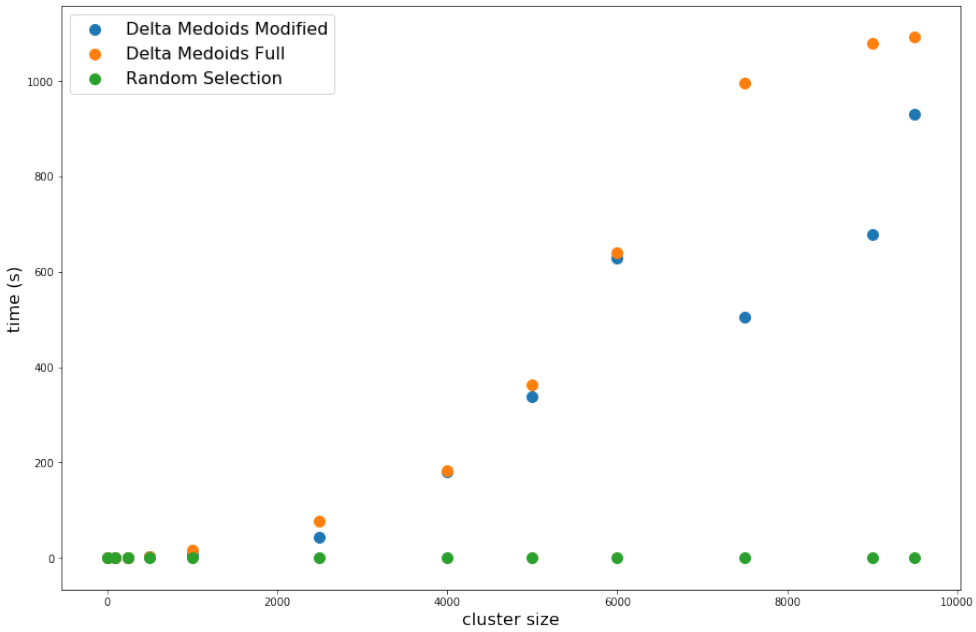


Figure 2.3: Comparing run times of $\delta$-Medoids and Random Selection

Chapter 3

# Datasets

The task of cluster representation in non-metric spaces generally is not necessarily connected only to the security field. Before moving to a dataset collected from a real network, the selected algorithms were first tested on artificially created datasets and well-labeled datasets found in the literature.

Below, the datasets used in this thesis are presented. Three types of datasets were used: artificially created, image recognition and real network security dataset. Each section corresponds to one of the listed dataset types.

## 3.1 Datasets Created for this Work

These datasets were created to test the specific properties of tested methods. They were made by using the Scikit-Learn library for Python 3 [13]. The datasets created are the following:

- Blobs 3D
- Overlap
- Circles 3D
- Moons

All of these datasets are in a metric space of two or three dimensions. They can be easily visualized on graphs and show some of the specific properties of implemented algorithms.

The Blobs 3D dataset consists of three clusters that are scattered randomly around a center point. This dataset was selected to test what samples will be selected by methods explained in Chapter 2 in clusters that can be easily represented by a centroid. The Overlap dataset consists of 3 clusters in shapes of blocks. These blocks overlap partly to test how selected algorithms cope with overlapping clusters. Circles 3D and Moons are one of the hardest

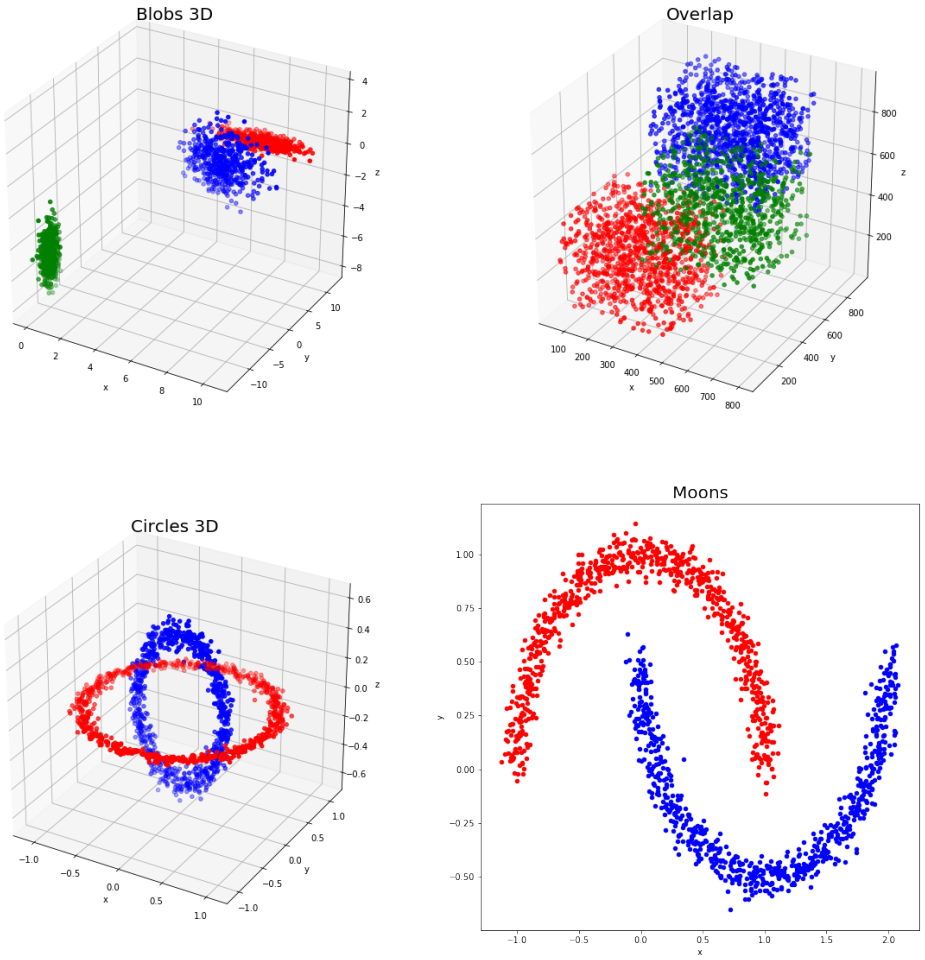


Figure 3.1: Graphs of all datasets

datasets to represent even by methods that are designed specifically for the metric space. These two datasets were chosen to test the resistance to noise in selecting representatives. All of these datasets are shown in Figure 3.1.

## 3.2 Image Recognition Datasets

One of the best-documented application domains for clustering is image recognition. Therefore, two datasets from this area were chosen for measuring the precision of each method. The first one is a labeled dataset of features collected from handwritten digits. The other one consists of annotated black and white pictures of clothing, shoes, and handbags.

The Pen-Based Recognition of Handwritten Digits Data Set [14] consists of 10922 samples. A sample represents 16 features collected from a handwritten digit. For each digit, there is a little more than a thousand samples. Features represent coordinate information about the digits as they were written on a $500 \times 500$ pixel frame. This dataset is split to train dataset (70%) and test dataset (30%).

The MNIST Fashion dataset [15] consists of a training set of 60,000 samples and 10,000 test samples. A sample from this dataset is a grayscale image of a piece of clothing, a shoe or a handbag. Each sample belongs to one of 10 classes. This problem is more complicated than handwritten digits from the previous dataset. It is a current benchmark dataset for classification problems. The MNIST Fashion dataset was introduced quite recently, in 2017.

## 3.3 Network Security Dataset

Real network traffic captured from a company with approximately 20 thousand employees. The capture is from more than 24 hours of a working day and consists of 33.5 million flows. Hosts that appeared in that capture were clustered using the method explained in Section 2.2. There are 16 clusters with sizes ranging from less than a hundred to several thousand. These clusters with all hosts labeled were used as real network dataset in this thesis. Labels provided for this dataset are the output of community-based clustering in Cognitive Targeted Anomaly Detection Framework. Similarity threshold used for the clustering was 0.8.

Each sample represents one device on the network. For such a big network the behaviors of devices are very diverse. This leads to heterogeneous clusters with behavior that changes over time. Only a pair-wise similarity between each host pair exists. Representation of cluster created is a complex problem, suitable for testing the properties of algorithms selected in this thesis.

Chapter 4

# Experiments

In this chapter, all performed experiments are described. Each section represents one experiment with its motivation, experimental setup and results.

All experiments were made in Jupyter Notebook technology using Python 3.7 kernel. For loading data from files and storing them in memory the Pandas library version 0.23.4 was used [16].

There are two main criteria for evaluating the result of each experiment corresponding with the definition of the problem in Section 2.3.1. One is the number of selected samples. The other one is the coverage provided by these representatives. The coverage represents the percentage of samples from the whole cluster that are closer than $\delta$ to one of the selected representatives. A table with these two measures is presented for each experiment.

For testing the ability to assign new samples to its corresponding cluster a part of each dataset was separated as test data. The algorithms $K$-Nearest Neighbors with $K = 1$ was then used to classify the test data based on the cluster prototypes created by selected algorithms. For each dataset, a confusion matrix was created to visualize the results of the classification test.

## 4.1 Test on Visualizable Datasets

The first experiment was run on the visualizable datasets created explicitly for this thesis. Seeing clustering results on visualizable datasets provides insight into algorithms performance.

The methods that were explained in Section 2.4 are usable on metric space with a distance measure as metric space is a special case of a topological space for which they are designed. Using them on visualizable benchmark datasets designed to address specific problems helped to determine their properties such as how many representatives each algorithm selects and how well does the representatives work in classification in newly added samples.

### 4.1.1 Experimental Setup

Datasets used for this experiment were:

- Blobs 3D
- Overlap
- Moons
- Circles 3D

All five algorithms were run. The $\delta$ was estimated as 5% quantile of distances from one random point in the data to all of the others. This choice was made because the desired number of samples selected is around 20. The similarity measure used in these datasets was Euclidean distance.

This experiment was performed in five runs using cross-validation. Each dataset was split into five 20% parts. In each run, four parts were combined to form the training data, and the remaining part was used to simulate newly added samples to be classified to their corresponding cluster.

### 4.1.2 Results

The results for each dataset are listed below. A table with the number of representatives selected from each cluster is presented as well as confusion matrix with correct and incorrect classifications of each test sample.

**Blobs 3D**

| Cluster | Training Samples | Greedy Selection | $\delta$-Medoids One-Shot | $\delta$-Medoids Full | $\delta$-Medoids Modified | Random Selection |
|---|---|---|---|---|---|---|
| A | 400 | 21 (100%) | 23 (100%) | 25 (100%) | 16 (96.75%) | 16 (86.25%) |
| B | 400 | 17 (100%) | 20 (100%) | 23 (100%) | 20 (99%) | 20 (92.75%) |
| C | 400 | 19 (100%) | 20 (100%) | 22 (100%) | 20 (99.25%) | 20 (92%) |

Table 4.1: Number of selected representatives with coverage in percent for the Blobs 3D dataset

For the Blobs dataset, which is a straightforward representation problem, all algorithms were able to cover the dataset almost entirely. Lower coverage for Random Selection can be seen, but the density and distribution of samples still make it cover most of the cluster. It is expected of the $\delta$-Medoids Modified not to cover the extreme borders of the dataset which are expected to represent up to 5% of the dataset. Even though the coverage is not 100%, the clusters are represented sufficiently for new samples to be classified correctly. It is a useful feature, as the algorithm is not expected to cover samples that are too distant from the rest of the cluster.

The number of representatives selected is approximately 20 that highly corresponds with the choice of $\delta$. This is not a problem as the number is still

relatively small (less than 10% of the dataset). Also, in Cognitive Targeted Anomaly Detection Framework, clusters do dynamically change, and around 20 representatives are requested to keep track of the cluster over time.

The confusion matrix of classification test can be seen in Figure 4.1.

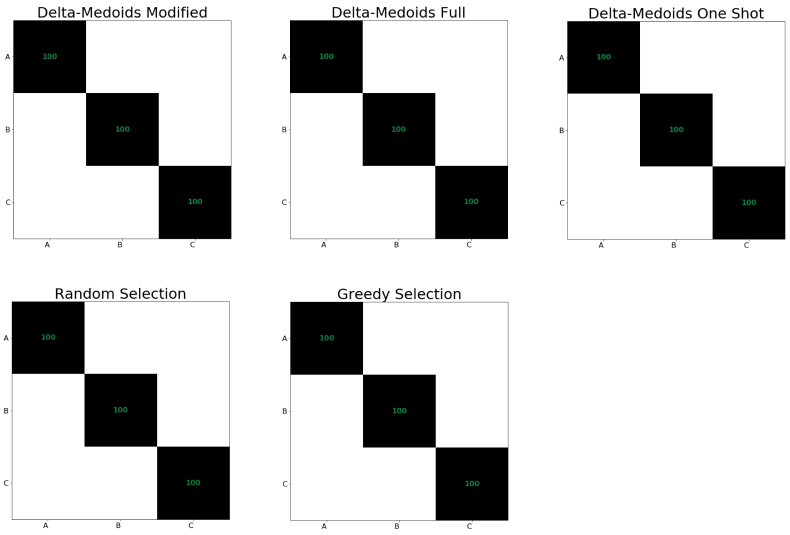


Figure 4.1: Confusion matrices for Blobs 3D dataset

### Overlap

| Cluster | Training Samples | Greedy Selection | $\delta$-Medoids One-Shot | $\delta$-Medoids Full | $\delta$-Medoids Modified | Random Selection |
|---|---|---|---|---|---|---|
| A | 800 | 25 (100%) | 23 (100%) | 23 (100%) | 23 (99.75%) | 23 (87.88%) |
| B | 960 | 26 (100%) | 21 (100%) | 23 (100%) | 21 (100%) | % 21 (81.15%) |
| C | 640 | 22 (100%) | 24 (100%) | 24 (100%) | 24 (99.84%) | 24 (88.6%) |

Table 4.2: Number of selected representatives with coverage in percent for the Overlap dataset

In the Overlap dataset, the average fit of the data remains similar to the Blobs 3D dataset as it is calculated for each cluster separately. Therefore, the number of representatives selected remains determined only by choice of $\delta$.

The results of the classification test show that classifying border samples from overlapping clusters is not a simple task. Confusion matrices present slightly better results with the representatives selected by $\delta$-Medoids modified. However, the number of misclassified samples for each dataset is not significant

when compared to the number of samples in the cluster. The confusion matrix can be seen in Figure 4.2.

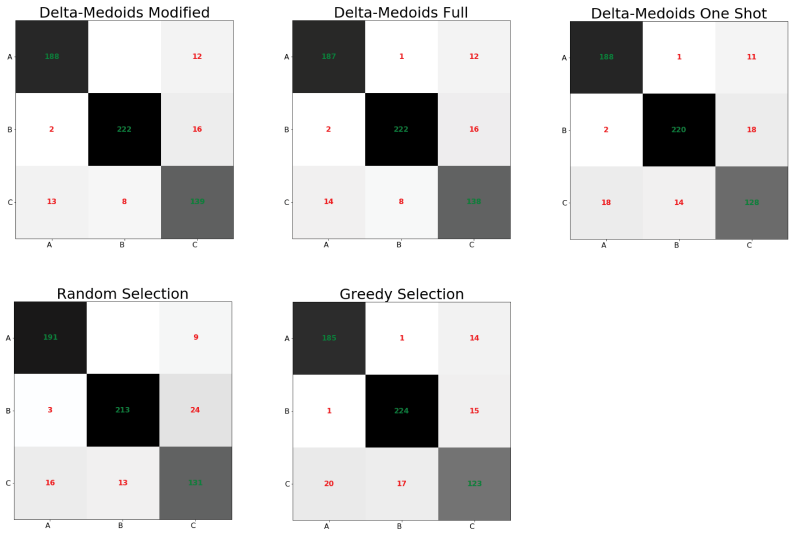


Figure 4.2: Confusion matrices for Overlap dataset

### Moons and Circles 3D

| Cluster | Training Samples | Greedy Selection | $\delta$-Medoids One-Shot | $\delta$-Medoids Full | $\delta$-Medoids Modified | Random Selection |
|---|---|---|---|---|---|---|
| A | 600 | 22 (100%) | 21 (100%) | 22 (100%) | 21 (100%) | 21 (88.67%) |
| B | 600 | 21 (100%) | 24 (100%) | 25 (100%) | 24 (100%) | 24 (67%) |

Table 4.3: Number of selected representatives with coverage in percent for the Moons dataset

| Cluster | Training Samples | Greedy Selection | $\delta$-Medoids One-Shot | $\delta$-Medoids Full | $\delta$-Medoids Modified | Random Selection |
|---|---|---|---|---|---|---|
| A | 600 | 15 (100%) | 13 (100%) | 13 (100%) | 13 (100%) | 13 (87.5%) |
| B | 600 | 6 (100%) | 8 (100%) | 8 (100%) | 8 (99.83%) | 8 (83.5%) |

Table 4.4: Number of selected representatives with coverage in percent for the Circles 3D dataset

Selecting representatives both Moons and Circles 3D datasets is a harder problem than for the compact clusters in Blobs 3D and Overlap datasets.

Clusters in the Moons dataset are not easily represented by one centroid. The results show that medoid selection is a good option for representing this type of clusters. Each method, excluding Random Selection, was able to cover the whole cluster. Randomly selected representatives suffer from the uneven distribution of samples in the space, thus covering a lower portion of the cluster. On these two datasets, the Random Selection algorithm performs worse as can be seen in the corresponding confusion matrices. The confusion matrix for the Moons dataset can be seen in Figure 4.3, for the Circles 3D dataset in Figure 4.4

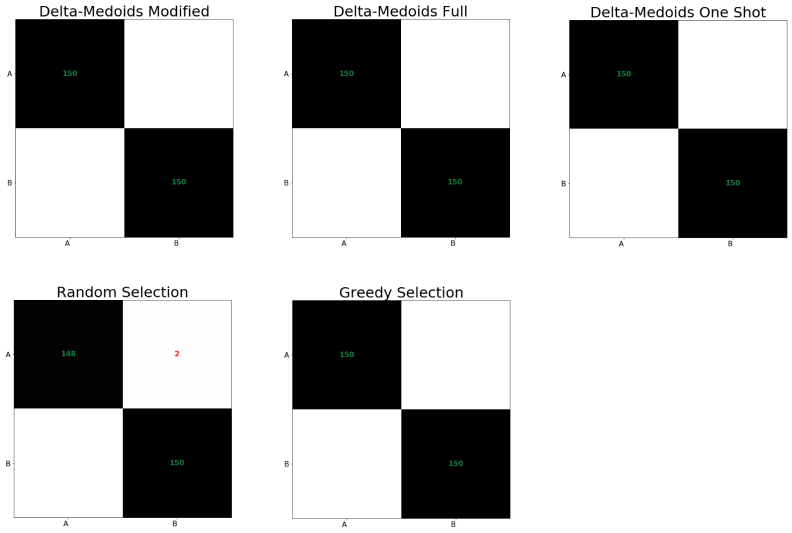


Figure 4.3: Confusion matrices for Moons dataset

From the results listed the difference between $\delta$-Medoids Full and Modified are not easily distinguishable. The difference between these two algorithms can be seen in Figure 4.5. The full version of the algorithm tends to select the border samples of a dataset as representatives. In these datasets, the results do not show this because they are relatively dense and have a normal distribution.

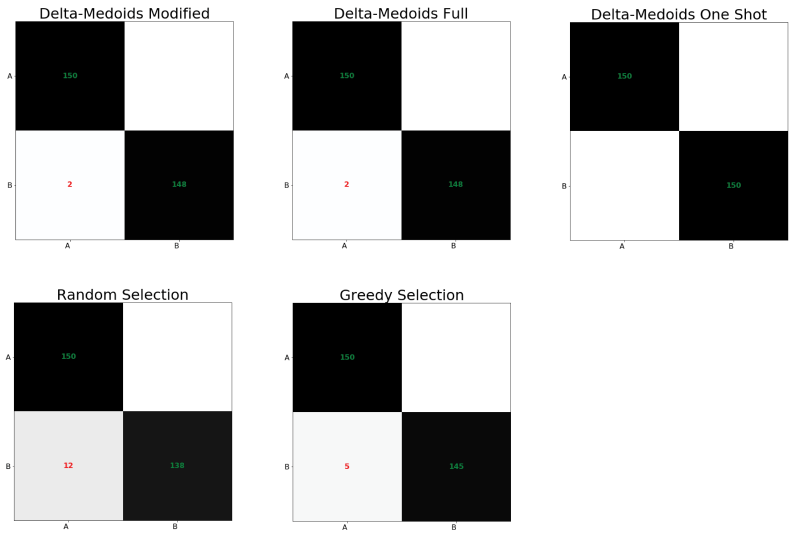


Figure 4.4: Confusion matrices for Circles 3D dataset

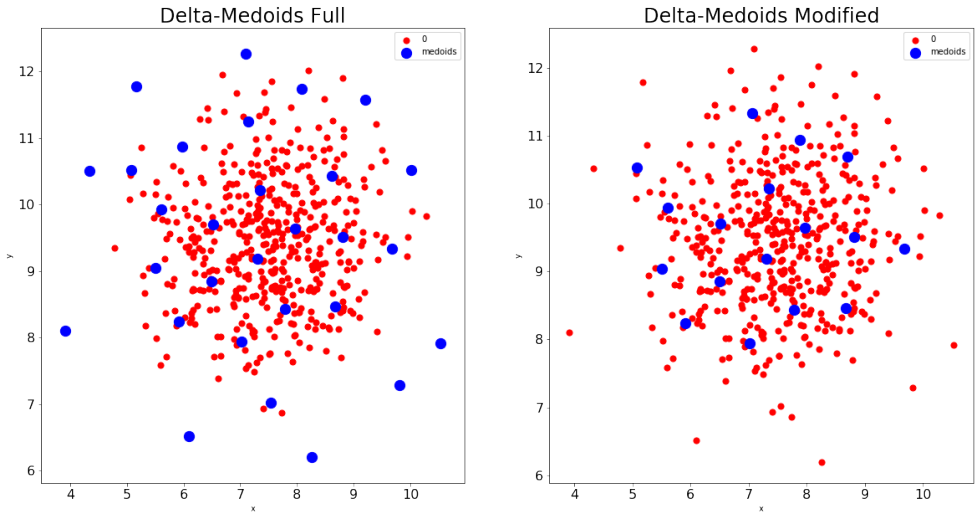


Figure 4.5: Difference between $\delta$-Medoids Full and Modified

## 4.2 Test on Image Recognition Datasets

The motivation for this experiment was to test the algorithm results on well-explored and well-annotated datasets commonly used as a benchmark for clustering. Samples in these datasets are distributed more sparsely, and the clusters overlap. This experiment intended to see the difference in numbers of representatives selected by the $\delta$-Medoids One-Shot and the $\delta$-Medoids Modified algorithms. Also, how this difference influences the classification tests.

### 4.2.1 Experimental Setup

Datasets used for this experiment were:

- Pendigits
- MNIST Fashion

In this and the following experiments, the $\delta$-Medoids Full algorithm was not used, and only the Modified algorithm was tested. The $\delta$ was estimated as 5% quantile of distances from one random point in the data to all of the others. The similarity measure used in these datasets was Euclidean distance.

### 4.2.2 Results

The results for different methods are listed below.

**Pendigits**

| **Cluster** | **All Samples** | **Greedy Selection** | **$\delta$-Medoids One-Shot** | **$\delta$-Medoids Modified** | **Random Selection** |
|---|---|---|---|---|---|
| 0 | 780 | 33 (100%) | 33 (100%) | 33 (100%) | 33 (93.46%) |
| 1 | 779 | 33 (100%) | 41 (100%) | 41 (100%) | 41 (94.48%) |
| 2 | 780 | 17 (100%) | 15 (100%) | 15 (100%) | 15 (95.71%) |
| 3 | 719 | 12 (100%) | 9 (100%) | 9 (100%) | 9 (94.64%) |
| 4 | 780 | 25 (100%) | 24 (100%) | 24 (100%) | 24 (93.59%) |
| 5 | 720 | 25 (100%) | 27 (100%) | 27 (100%) | 27 (93.89%) |
| 6 | 720 | 16 (100%) | 16 (100%) | 16 (100%) | 16 (97.3%) |
| 7 | 778 | 22 (100%) | 20 (100%) | 20 (100%) | 20 (94.24%) |
| 8 | 719 | 75 (100%) | 77 (100%) | 15 (100%) | 15 (91.24%) |
| 9 | 719 | 50 (100%) | 52 (100%) | 13 (100%) | 13 (92.65%) |

Table 4.5: Number of selected representatives with coverage in percent for the Pendigit dataset

The results for Pendigits dataset show that the $\delta$-Medoids algorithm was able to represent the whole dataset with perfect precision. The Random Selection lacks behind in percentage. The confusion matrix in Figure 4.6 shows that the clusters partly overlap (i.e., digits 1 and 7). Even with these overlaps, the

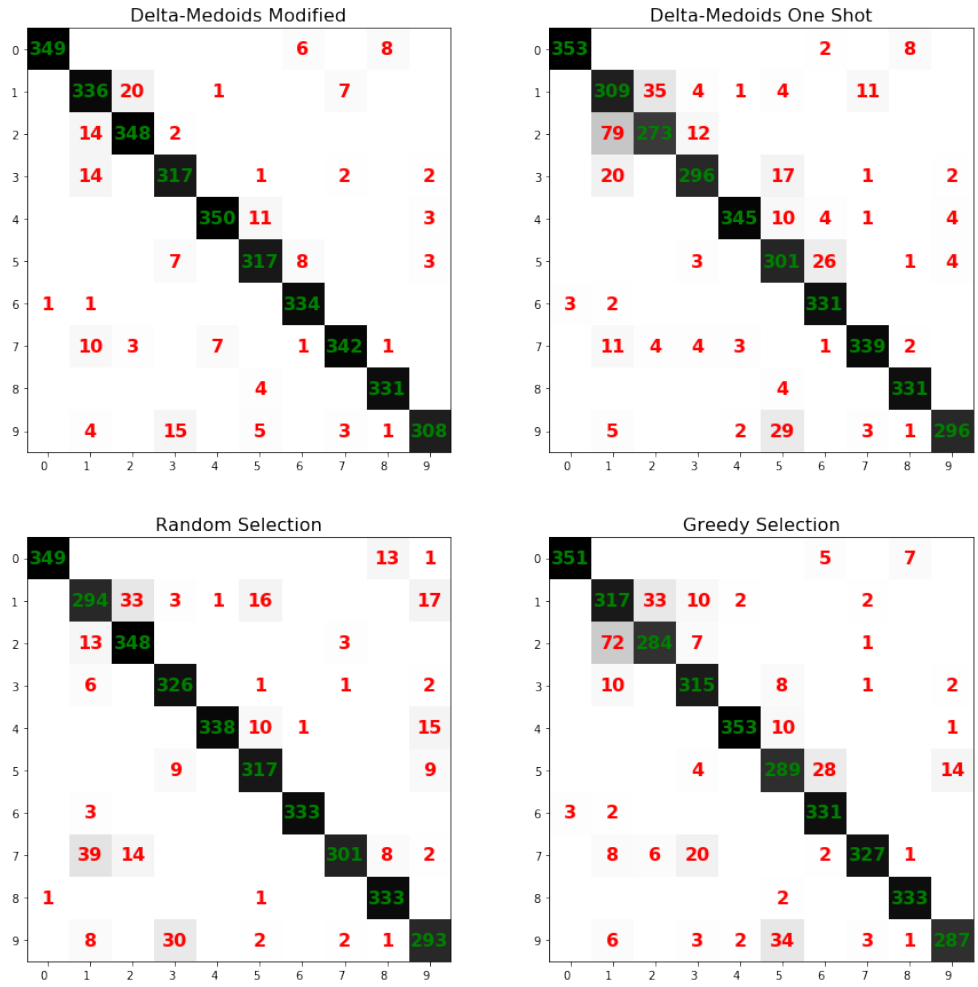


Figure 4.6: Confusion matrices for Pendigits dataset

$\delta$-Medoids algorithm works best for classification. It selects the lowest number of representatives while getting the best results.

The Greedy Selection and $\delta$-Medoids One-Shot select the same number of representatives, and their results are also very similar. This is interesting as the complexity of these algorithms differs a lot, Greedy Selection being much slower with $\mathcal{O}(n^2)$.

### MNIST Fashion

In the experiment for this dataset, the Greedy Selection algorithm was omitted as the tests would take too long on clusters that contain 6000 samples. Previous experiments show that the $\delta$-Medoids algorithm chooses a similar amount of representatives while getting better coverage.

| Cluster | All Samples | $\delta$-Medoids One-Shot | $\delta$-Medoids Modified | Random Selection |
|---|---|---|---|---|
| 0 | 6000 | 859 (100%) | 23 (100%) | 23 (93.41%) |
| 1 | 6000 | 176 (100%) | 16 (100%) | 16 (95.58%) |
| 2 | 6000 | 845 (100%) | 20 (100%) | 20 (93.37%) |
| 3 | 6000 | 616 (100%) | 22 (100%) | 22 (95.33%) |
| 4 | 6000 | 630 (100%) | 25 (100%) | 25 (94.98%) |
| 5 | 6000 | 1622 (100%) | 14 (100%) | 14 (94.36%) |
| 6 | 6000 | 1181 (100%) | 18 (100%) | 18 (95.06%) |
| 7 | 6000 | 266 (100%) | 19 (100%) | 19 (95.12%) |
| 8 | 6000 | 1971 (100%) | 16 (100%) | 16 (94.72%) |
| 9 | 6000 | 686 (100%) | 28 (100%) | 28 (93.56%) |

Table 4.6: Number of selected representatives with coverage in percent for the MNIST Fashion dataset

The coverage for the MNIST Fashion shows that estimation of the $\delta$ as the 5% quantile of distances from one random sample to the rest works well even for a very complex dataset.

However, the complete coverage did not ensure accurate results it the classification test. For each method the number of correctly classified samples out of 10000 is listed below as the difference between algorithms is not noticeable on first sight from the confusion matrix in Figure 4.8. It represents the sum of numbers on the diagonal of the matrix. For Random Selection the number of correctly classified samples is 6520, meaning 65.5% of test samples were classified correclty. For $\delta$-Medoids One-Shot it is 6826 (68.26% classified correctly) and for $\delta$-Medoids Full it is 7053 (70.53% classified correctly). These numbers confirm that selecting the correct representatives can improve the classification results, especially when the Modified and One-Shot version of $\delta$-Medoids are compared. The Modified algorithm was able to get better results with a much lower number of representatives. In some cases the number of representatives selected is for $\delta$-Medoids Modified is less than 1% of the number selected with the One-Shot version.

## 4.3 Test on Network Security Data

This experiment tests the functionality of selected methods as if they were plugged into the toolchain of Cognitive Targeted Anomaly Detection described in Section 1.4. The data for it was collected from the framework.

### 4.3.1 Experimental Setup

Clustered hosts from a real network were used for this experiment. This dataset is explained in greater detail in Section 3.3. From this dataset traffic

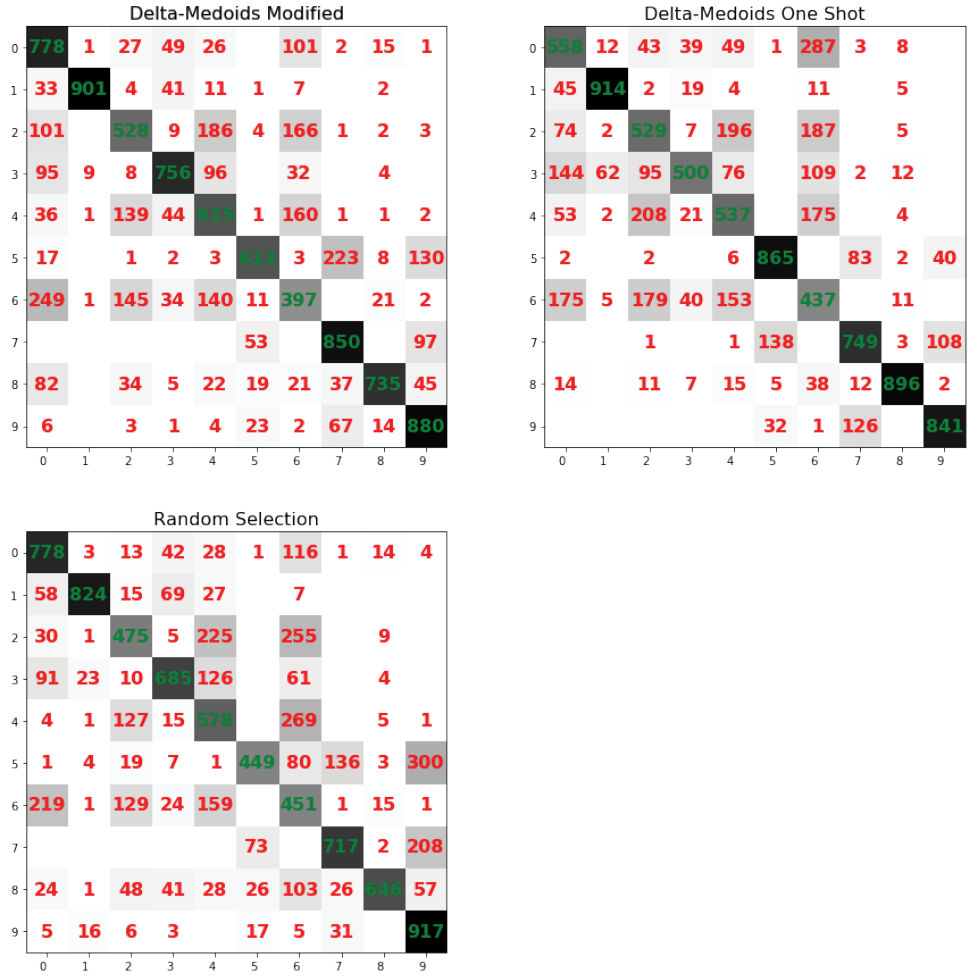


Figure 4.7: Confusion matrices for MNIST Fashion dataset

for 24 hours was used for training the model. Data from the following 4 hours was used for testing the representatives selected. The similarity measure from Section 1.4 was used.

Three algorithms were run, $\delta$-Medoids Full and Greedy Selection were omitted for reasons listed in the previous section. The $\delta$ was set to 0.8 as was the threshold used in community-based clustering for their creation.

### 4.3.2 Results

The results for different methods are listed below.

| Cluster | All Samples | $\delta$-Medoids One-Shot | $\delta$-Medoids Modified | Random Selection |
|---|---|---|---|---|
| A | 72 | 7 (100%) | 7 (100%) | 7 (90.27%) |
| B | 127 | 23 (100%) | 27 (100%) | 27 (90.55%) |
| C | 2263 | 1480 (100%) | 46 (28.94%) | 46 (8.13%) |
| D | 124 | 76 (100%) | 17 (56.45%) | 17 (54.03%) |
| E | 89 | 68 (100%) | 16 (41.57%) | 16 (38.2%) |
| F | 54 | 38 (100%) | 38 (100%) | 38 (87.04%) |
| G | 87 | 41 (100%) | 42 (100%) | 42 (80.46%) |
| H | 1149 | 503 (100%) | 34 (62.92%) | 34 (53.61%) |
| I | 60 | 53 (100%) | 12 (33.33%) | 12 (30%) |
| J | 77 | 73 (100%) | 14 (23.37%) | 14 (20.78%) |
| K | 2692 | 912 (100%) | 25 (66.12%) | 25 (51.98%) |
| L | 137 | 109 (100%) | 22 (36.5%) | 22 (33.57%) |
| M | 53 | 35 (100%) | 35 (100%) | 35 (77.34%) |
| N | 259 | 15 (100%) | 16 (100%) | 16 (98.07%) |
| O | 76 | 58 (100%) | 13 (39.47%) | 13 (34.21%) |
| P | 76 | 76 (100%) | 7 (9.21%) | 7 (9.21%) |

Table 4.7: Number of selected representatives with coverage in percent for the real network data

The coverage results differ significantly cluster to cluster. $\delta$-Medoids Modified still gets better coverage for each cluster (except for cluster P). However, a big decrease in coverage can be seen in both $\delta$-Medoids Modified and Random Selection algorithm. This reflects the nature of cluster creation in the modified algorithm. Many clusters from $\delta$-Medoids One shot, which represents the basic routine repeated in the Modified algorithm, are merged with the most similar neighbor. This neigbor can be less similar than the $\delta$ given (for this experiment $\delta = 0.8$).

The following numbers represent the percentage of samples from test data classified correctly. It is calculated by dividing the sum of diagonal of the confusion matrix by the number of samples in the test data. The confusion matrix can be seen in Figure 4.8 For $\delta$-Medoids One-Shot 77% of samples were classified correctly. Such a high percentage is given by the fact, that the algorithm sometimes selects even more than half of the dataset. For Random Selection this percentage is 63%. The $\delta$-Medoids Modified algorithm surpassed it at 79% of samples correctly assigned to their cluster in classification. Results in this paragraph show that even with low coverage the overall accuracy of classification is maintained. The $\delta$-Medoids Modified algorithm was able to get better classification results with much smaller coverage and representatives selected than the $\delta$-Medoids One-Shot. $\delta$-Medoids One-Shot algorithm shows

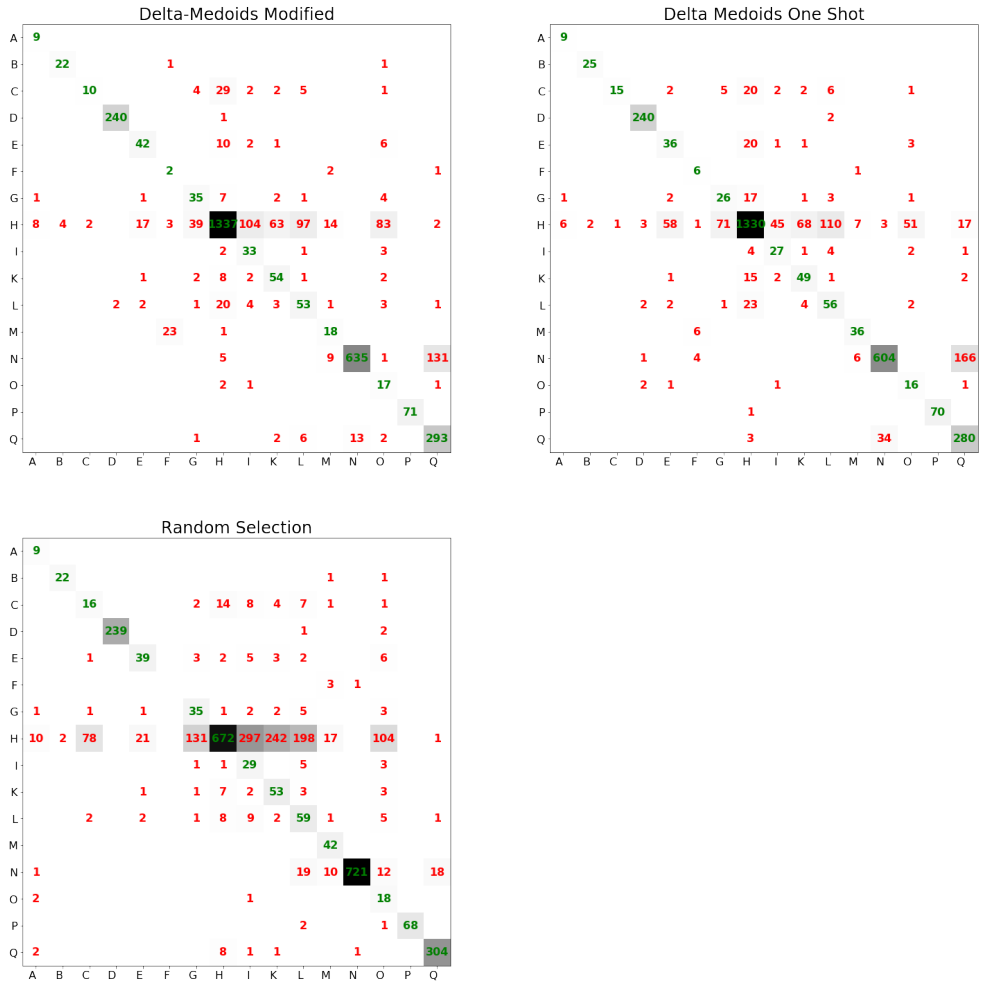


Figure 4.8: Confusion matrices for real network data

similar results in classification as the number of representatives selected by this algorithm is significantly higher containing on average 30% of each cluster.

A good improvement can be seen in results for cluster H. The $\delta$-Medoids algorithm was able to represent it the best with 1337 samples classified correctly. Even for clusters with several thousands of samples, few samples were selected a while retaining dominant properties for assigning points to clusters properly.

Another result worth noticing is that the $\delta$-Medoids Modified algorithm gives the means to find the number of representatives needed to represent the cluster. In random selection, a constant is needed which is hard to guess without prior knowledge about the cluster. For some clusters, 20 representatives might be enough, and for others, a higher number might be needed. Selecting

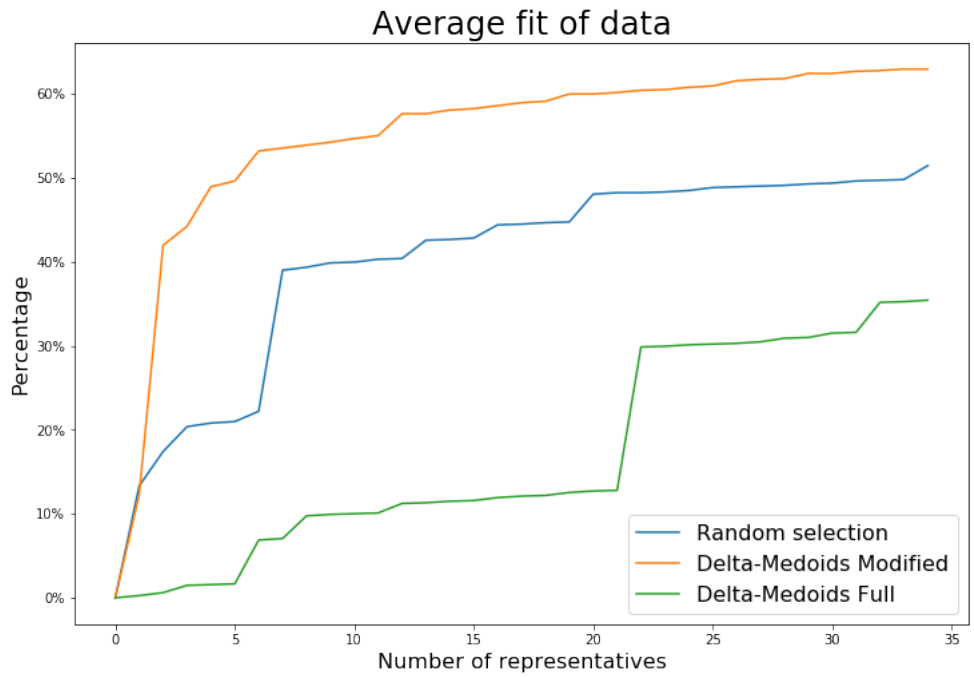


Figure 4.9: Average fit of representatives in one cluster

a higher number without justification could lead to unwanted computational overhead.

## 4.4 Fit of the Data

In this experiment the data fit as defined in Section 2.3 was tested. The motivation was to test how many samples are covered by each added sample and whether on average the $\delta$-Medoids algorithm performs better than Random Selection and $\delta$-Medoids Full algorithm on large clusters with high diversity.

### 4.4.1 Experimental Setup

In this experiment, one complex cluster (cluster K) from Network Security Dataset was represented by $\delta$-Medoids, $\delta$-Medoids Full and Random Selection algorithms. After the representatives were selected, the percentage of coverage by each representative was calculated and plotted into a graph.

### 4.4.2 Results

The results of this experiment are presented in Figure 4.9.

On average, the $\delta$-Medoids Modified algorithm covers the cluster better. It can be seen from the increasing distance of lines by each representative added. In some places, the slope of the Random Selection line is the same as of $\delta$-Medoids Modified. In these places, the algorithm selected a good representative by chance, similar to the ones chosen by a more sophisticated method. The line for $\delta$-Medoids Full shows that the algorithm chooses representatives from smaller subclusters. It does not mean that the algorithm does not fit the dataset well. Only the first 34 representatives are shown to match the other algorithms. The full algorithm selected 532 samples as representatives, which is approximately half of the dataset. With all of these representatives, the coverage would is 100%.

Results confirm that $\delta$-Medoids Modified algorithm achieves the best coverage of the cluster with the least samples selected as representatives.

# Conclusion

Methods that are relevant for creating cluster prototypes in a non-metric space were studied. Several methods including Random Selection, Greedy Selection and $\delta$-Medoids algorithms were chosen from prior art. Furthermore, a new algorithm $\delta$-Medoids Modified was introduced by modifying the $\delta$-Medoids Full algorithm. Implementation of each algorithm was written and tested on artificially created benchmark datasets. All algorithms were then further tested on image recognition datasets Pendigits and MNIST Fashion. Finally, the methods were fine-tuned and tested on a real Network Security Dataset captured in a medium-sized company.

The results presented in this thesis show that the modification made on $\delta$-Medoids Full improves its performance on network security datasets. The $\delta$-Medoids Modified algorithm can find a small number of representatives with decent coverage of the whole dataset. Furthermore, it runs faster than $\delta$-Medoids full while fulfilling criteria for representative selection. It is also able to choose the number of representatives based on the properties of the cluster rather than just guessing the right number of representatives, which is what happens in Random Selection. Representatives selected this way to fit the dataset on average better than selecting them randomly, which is the method currently used.

The method to represent clusters in non-metric space by a small subset of the whole cluster was implemented and tested in this thesis. This method was adjusted so that it can be incorporated into Cognitive Targeted Anomaly Detection Framework.

# Bibliography


[1] HRON, Martin. The 10 Biggest Data Breaches in 2018. In: *Avast Blog* [online]. Avast Software s.r.o., 2018. [2019-04-05]. Avaiable at: `https://blog.avast.com/biggest-data-breaches`

[2] MUKHERJEE, Biswanath; HEBERLEIN, L. Todd; LEVITT, Karl N. Network intrusion detection. *IEEE network*, 1994, 8.3: 26-41.

[3] COOPER, Stephen. 2019 Best Intrusion Detection Systems (10+ IDS Tools Reviewed). In: *comparitech* [online]. Comparitech Limited, 2019. [2019-04-27]. Available from: `https://www.comparitech.com/net-admin/network-intrusion-detection-tools/`

[4] LIAO, Hung-Jen, et al. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 2013, 36.1: 16-24.

[5] BROWNLEE, N.; MILLS, C.; RUTH, G. Traffic Flow Measurement: Architecture. RFC 2722, October 1999.

[6] GRILL, Martin. *Combining network anomaly detectors*. Praha, 2016. Doctoral Thesis. Czech Technical University in Prague. PEVNÝ, Tomáš; REHÁK, Martin.

[7] KOPP, Martin; GRILL, Martin; KOHOUT, Jan. Community-based anomaly detection. In: *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, 2018. p. 1-6.

[8] GUTTAG, John. *Introduction to Computation and Programming Using Python: With Application to Understanding Data*. Second. Cambridge, MA: MIT Press, 2016. ISBN 978-0-262-52962-4.

[9] CHOUDHARY, B. *The Elements of Complex Analysis*. New Age International, 1993. ISBN 9788122403992.

[10] STAHL, Saul and Catherine STENSON. *Introduction to Topology and Geometry* [online]. Somerset: John Wiley & Sons, Incorporated, 2014. ISBN 9781118108109.

[11] TAN, Pang-Ning; STEINBACH, Michael; KUMAR, Vipin. *Introduction to data mining*. Harlow: Pearson, 2014. ISBN 9781292026152.

[12] LIEBMAN, Elad; CHOR, Benny; STONE, Peter. Representative Selection in Nonmetric Datasets. *Applied Artificial Intelligence*, 2015, 29.8: 807-838.

[13] PEDREGOSA, Fabian, et al. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 2011, 12.Oct: 2825-2830.

[14] Dua, D.; Graff, C. UCI Machine Learning Repository. 2017. Available from: `http://archive.ics.uci.edu/ml`

[15] XIAO, Han; RASUL, Kashif; VOLLGRAF, Roland. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[16] MCKINNEY, Wes, et al. Data structures for statistical computing in python. In: *Proceedings of the 9th Python in Science Conference*. 2010. p. 51-56.

Appendix A

# Abreviations

**AD** Anomaly Detection

**DDoS** Distributed Denial of Service

**DPI** Deep Packet Inspection

**IDS** Intrusion Detection System

**NBAD** Network-Based Anomaly Detection

**NIDS** Network Intrusion Detection System

**RFC** Request for Comments

Appendix B

# Contents of attached USB disk

- `readme.txt` ..... the file with disk contents description
- `src` ..... the directory of source codes
  - `datasets` ..... the directory of datasets used in thesis
  - `experiments` ..... the directory of source codes of experiments
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format